Understandability Issues of Approaches Supporting Business Process Variability*

Victoria Torres[1], Stefan Zugal[2], Barbara Weber[2], Manfred Reichert[3], Clara Ayora[1], and Vicente Pelechano[1]

[1] Universitat Politècnica de València, Spain
{vtorres,cayora,pele}@pros.upv.es
[2] University of Innsbruck, Austria
{stefan.zugal,barbara.weber}@uibk.ac.at
[3] University of Ulm, Germany
manfred.reichert@uni-ulm.de

Abstract. The increasing adoption of Process-Aware Information Systems, together with the *reuse* of process knowledge, has led to the emergence of process model repositories with large process families, i.e., collections of related process model variants. For managing such related model collections two types of approaches exist. While *behavioral* approaches take supersets of variants and derive a process variant by hiding and blocking process elements, *structural* approaches take a base process model as input and derive a process variant by applying a set of change operations to it. However, at the current stage no framework for assessing these approaches exists and it is not yet clear which approach should be better used and under which circumstances. Therefore, to give first insights about this issue, this work compares both approaches in terms of understandability of the produced process model artifacts, which is fundamental for the management of process families and the *reuse* of their contained process fragments. In addition, the comparison can serve as theoretical basis for conducting experiments as well as for fostering the development of tools managing business process variability.

1 Introduction

The increasing adoption of Process-Aware Information Systems (PAISs) by industry has led to large collections of process models in a variety of application domains. Despite the particularities found in specific domains, many of these models share common parts of their definition (e.g., activities). We denote such related process variant models as *process family* in the following. By using common business process modeling languages such as BPMN, EPC, YAWL, or UML Activity Diagrams, process families can just be represented either in separate process models (one per process variant) or in a unified model using conditional

* This work has been developed with the support of MICINN under the project EVERYWARE TIN2010-18011.

branching for describing differences and subprocesses for capturing commonalities [1]. Even though the use of subprocesses improves the reuse of process fragments within the process family, this is not sufficient to capture variability. In particular, dependencies between process fragments as well as context information cannot be represented explicitly in the model, aggravating model maintenance.

To properly handle large process families (i.e., avoid redundancies, foster reusability, and reduce modeling efforts) several proposals have been developed in recent years (e.g., Provop [1], C-EPC [2], PESOA [3], Rule representation and processing [4], and PPM [5]), which can be classified as either *behavioral* or *structural* approaches. *Behavioral* approaches represent all members of the family within the same model artifact, capturing both, commonalities and particularities of all process variants. In turn, *structural* approaches use different artifacts to represent the family, e.g., by using a base model to which structural changes such as inserting, deleting, or moving activities may be applied to derive process variants.

To foster reusability of process model families, understandability of the created artifacts is essential. So far, however, no experimental insights regarding *quality* aspects (e.g., *understandability*) are available and it is not clear under which circumstances the use of one approach is more appropriate than the other. Since the results of assessing a process model's understandability significantly depend on the specific understandability tasks [6, 7], we have structured the comparison of both approaches along several such tasks using a process family from the film industry. For each of the tasks, the paper discusses the process followed by a model reader to accomplish such tasks using both approaches. In addition, we use cognitive psychology as a tool for explaining the differences between the two approaches. This comparison will provides us the theoretical basis for conducting experiments as well as for fostering the development of tools for managing variability in business processes.

The remaining sections are structured as follows. Section 2 presents a process family from the film industry along which the two different approaches are compared. Section 3 provides basic notions and introduces the *behavioral* and *structural* approaches for dealing with variability in business processes. Then, Section 4 describes the concepts from cognitive psychology that are used to assess understandability aspects of both approaches. Based on these concepts, Section 5 provides a qualitative comparison for a set of comprehension tasks regarding the understandability of the modeling artifacts built by each approach. Section 6 then presents an overview of related work. Finally, Section 7 concludes the paper and gives an outlook.

2 Example of a Process Family

As running example, we consider a modified version of a process family from the film industry for editing a screen project, which varies depending on the shooting media and delivery media used [8]. First, footage is received, either in *tape*, *film*,

or *tape and film* and *prepared for edition* depending on the shooting medium. Then *offline edition* is performed. Next, the cut stage is performed through *online edit* (if the *shooting medium* is *tape*), through *negmatching* (in the case of *film*), or through both cuts (when *shooting media* is *tape and film*). After this point, the finishing on a delivery medium phase starts. For this purpose, the delivery media (e.g., *tape, film, tape and film, or new medium*) must be chosen. As a result, different variants exist. For example, when shooting media is *film* or *tape and film* and cutting has been performed through negmatching, a *finish on film* has to be performed to maintain the quality of the delivery medium. On the contrary, if the cutting is performed through *online editing* and *film* or *tape and film* is the delivery medium, *record digital film master* needs to be mandatorily performed to transfer the editing results to *film*. Similarly, *telecine transfer* will be performed only if negmatching is performed previously and the expected delivery format is *tape* or *new medium*. Finally, if neither *tape* or *film* have been chosen activity *finish on new medium* must be performed.

3 Approaches for Modeling BP Variability

To properly represent variability in a BP model, it is important to define (1) what parts of the BP model may vary according to a specific context, (2) what alternatives exist in each of those parts, and (3) which conditions make these alternatives being selected. The first issue refers to the precise identification of the parts being subject to variation, which are commonly known as *variation points* (e.g., the edition phase which depends on the type of medium used for footage—*tape*, *film*, or *tape and film*). The second issue refers to the different alternatives that exist for all those variation points (e.g., the alternative activities for the edition phase are *edit footage online* (when shooting medium is *tape*) and *perform negmatching* (when shooting medium is *film*)). In addition, some models may require the definition of *relationships* (e.g., inclusion, exclusion) between alternative process fragment from different variation points (e.g., activity *record digital film master* has to be performed when *negmatching* has been performed and the *delivery medium* is *film* or *tape and film*). The third issue refers to the *context* of these variations, which is usually represented by a set of variables gathered in a *context model* in which the BP model is used (e.g., *shooting* and *delivery media* types). These issues are not addressed by common business process modeling languages (e.g., BPMN, EPC, YAWL) and their consideration turns crucial to properly represent and manage process families. The following subsections present two different approaches targeted at the representation of such families, i.e., *behavioural* and *structural*.

3.1 Behavioural Approach

The behavioral approach represents a process family in a single artifact, known as *configurable process model* capturing both the commonalities and particularities

of the *process variants* reflecting all possible behavior. Proposals that have been developed following this approach include C-EPC [2], C-YAWL [9] or Feature-EPC [10]. In the following we take C-EPC [2] as the representative proposal since it constitutes the most well-known and mostly cited proposal in literature. C-EPC extends EPC with configurable elements (i.e., *configurable nodes* and *configuration requirements*) to explicitly model variability. Fig. 1 illustrates the configurable process model representing the postproduction process (cf. Section 2). On the one hand, *configurable nodes* (i.e., functions and connectors) are represented graphically with thick solid borders and define variations points in the model where different alternatives may exist. Specifically, *configurable functions* (e.g., activity *telecine transfer* in Fig. 1) can be configured as ON (i.e., function is kept in the model), OFF (i.e., function is removed from the model), or OPT (i.e., conditional branching is included in the model deferring the decision to run-time). *Configurable connectors*, in turn, can be configured to an equally or more restrictive connector. For example, a configurable OR can be configured as a regular OR (not applying any restrictions), or can restrict its behaviour by configuring it as an XOR (i.e., selecting one of the outgoing/incoming alternatives), AND (i.e., selecting all of the outgoing/incoming alternatives), or SEQ_n (i.e., reducing the alternatives for the configuration of the connector to just one of its outgoing/incoming sequences). Table 1 summarizes all possible configurations for the different types of connectors (default configurations, i.e., configurations applied when no constraints are defined, are marked with asterisks).

Config. connector	OR	XOR	AND	SEQ_n
OR	X*	X	X	X
XOR		X*		X
AND			X	

Table 1. Allowed configurations of *configurable connectors*

On the other hand, *configuration requirements* are graphically represented as tags attached to *configurable nodes* and formalize, by means of logical predicates, domain constraints related to the attached *nodes*. The configuration of the node will the be made based on the evaluation of the attached *configuration requirements*. For example, *requirement 5* states that configurable connector 3 has to be configured as $SEQ3_a$ when configurable connector 1 has been configured as $SEQ1_a$, i.e., activity *edit footage online* has to be performed when shooting medium is *tape*, forbidding implicitly the execution of activity *perform negmatching*). However, *configurable nodes* not always have requirements attached to them (since they are only included when there is a constraint regarding their configuration). When no requirements are attached to a configurable node, this should be transformed into a regular one, maintaining the behaviour of the original connector and deferring the configuration decision to run-time.

Note that some requirements (i.e., req. 1–3 and 9–12) comprise context variables, providing information regarding the context that drives the configuration of the associated configurable nodes, while others (i.e., req. 4–8 and 13–16) are expressed in terms of the structure of the model only. In the original proposal [8] only requirements of the latter type are used and the context variables only become explicit when using the proposal in combination with the questionnaire approach [32]. To make the C-EPC model better understandable we added requirements of the former category to make the context for choosing a certain configuration explicit.

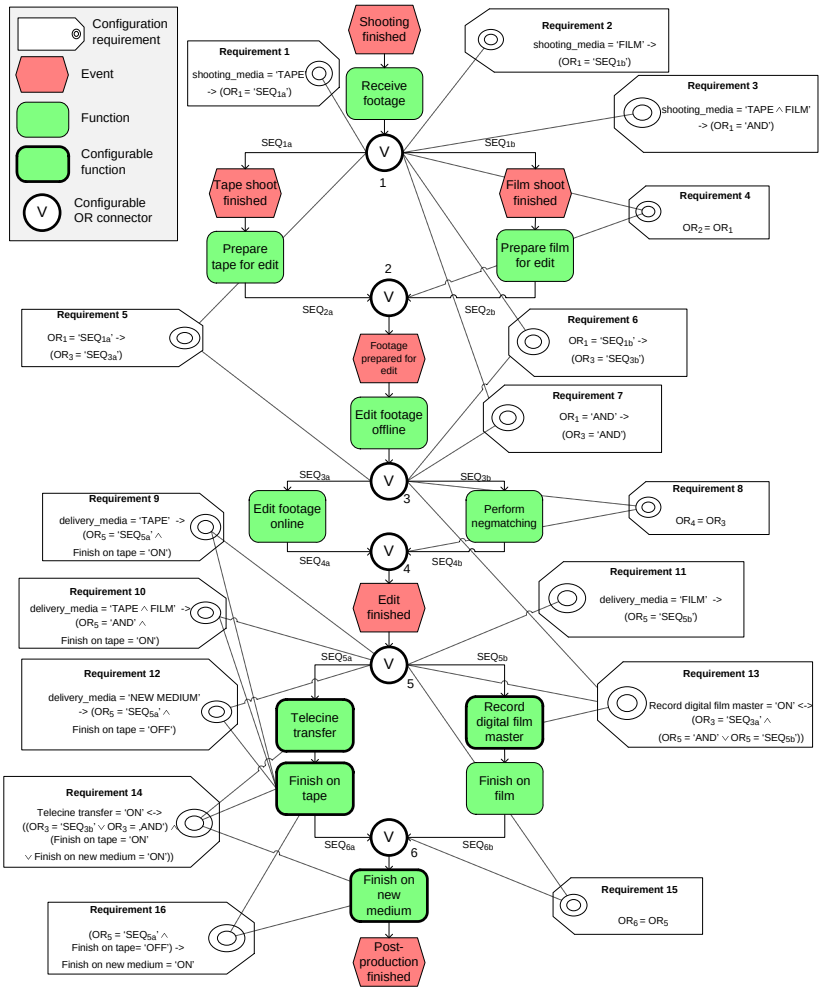

Fig. 1. C-EPC model for the screen postproduction process

3.2 Structural Approach

This approach proposes a gradual construction of the process family by modifying the structure of a specific process variant (called *base model*) at specific points (i.e., *variation points*) through *change operations*. Following this approach, we find proposals such as Provop [1] or Rule representation and processing [4]. In the following we use Provop as representative for the structural approach, since it can be considered the most widely used proposal for this approach. Fig. 2 illustrates the Provop proposal using the screen postproduction process (cf. Section 2). Provop allows creating process variants by adjusting the *base model* (cf. top part of Fig. 2) by the application of a set of high-level change operations between a couple of *adjustment points*. Depending on the modeling situation and on the existing process landscape, the construction of the *base model* can differ. [11] presents different policies (i.e., standard process, most frequently used process, minimal average distance, superset of all process variants, and intersection of all process variants) to perform such construction. Furthermore, Provop allows for more complex configuration adjustments by grouping multiple *change operations* into so-called *change options* (e.g., option 1 combines 2 delete and 2 insert operations). *Change options* are associated with a *context rule*, which is used at configuration time to decide, whether a certain option is applicable for the given context (e.g., regarding option 1 the context rule states that this option should only be applied if *shooting media* is *tape*). In addition, Provop allows for an explicit representation of different dimensions of the context (i.e., *context variables* and allowed *values*) by means of the *context model* (cf. bottom left part of Fig. 2). Finally, to prevent the derivation of semantically invalid variants, Provop provides the *constraint model* (cf. bottom right part of Fig. 2) which allows defining *inclusion*, *exclusion*, *order of application*, *hierarchy*, and *cardinality* relationships between *change options* (e.g., the application of *option 1* excludes the application of *option 2*).

4 Concepts from Cognitive Psychology

In order to discuss differences between C-EPC and Provop in Section 5, we will make use the concepts of *external memory*, *abstraction*, and *split-attention effect* from cognitive psychology as introduced in the following and transfer them to the domain of BP variability.

Basically, three different problem-solving "programs" or "processes" are known from cognitive psychology: search, recognition, and inference [12]. Search and recognition allow identifying information of rather low complexity, i.e., locating an object or recognizing patterns. Most models, however, go well beyond complexity that can be handled by search and recognition. For instance, a Boolean expression certainly cannot be interpreted just by looking at it and without deliberate thought. Here, the human brain as *"truly generic problem solver"* [13] comes into play. Thereby, cognitive psychology differentiates between working memory that contains the information is currently processed, as

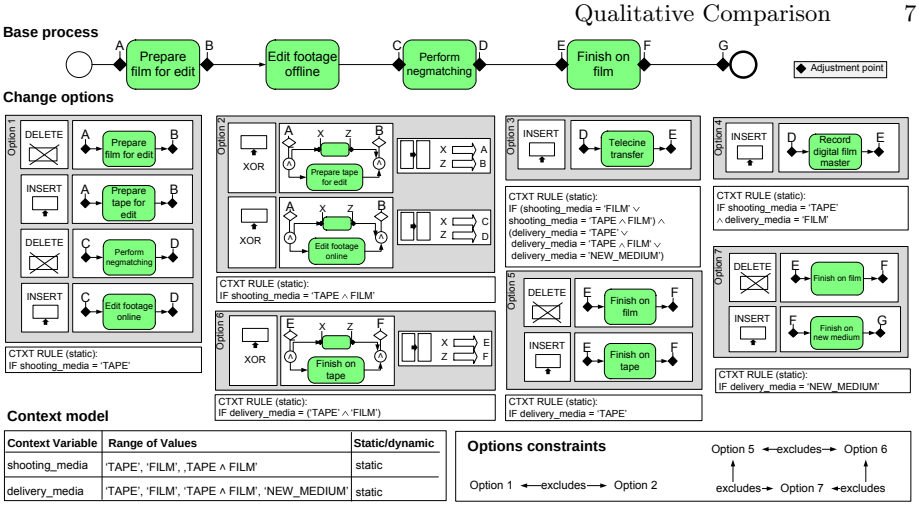

Fig. 2. Provop model for the screen postproduction process

well as long-term memory in which information can be stored for a long period of time [14]. Most severe, and thus of high interest and relevance, are the limitations of the working memory. As reported in [15], working memory cannot hold more than 7±2 items at the same time. In this context, the concept of mental effort, i.e., the amount of working memory used, is of interest, as it can be used to assess understanding. In addition, information held in the working memory decays after 18–30 seconds if not rehearsed [13]. The importance of the working memory has been recognized and led to the development and establishment of Cognitive Load Theory, meanwhile widespread and empirically validated in numerous studies [36]. The concept of mental effort, i.e., the amount of working memory used, is of interest, as it can be used to assess understanding. As discussed in [16], higher mental effort is in general associated with lower understanding, or more generally, errors are more likely to occur when the working memory's limits are exceeded [17]. Instead of only measuring accuracy or time for assessing the understandability of a notation, measuring mental effort allows for a more fine-grained analysis. In particular, when differences with respect to understandability are small, accuracy may not change, even though the mental effort changes significantly (cf. [18]). Subsequently, we will discuss three factors that influence the required mental effort: *external memory*, *abstraction*, and the *split-attention effect*.

External Memory. First, we would like to introduce a mechanism known for reducing mental effort, i.e., the amount of working memory slots in use. An external memory is referred to any information storage outside the human cognitive system, e.g., pencil and paper or a blackboard [17, 13, 19, 12]. Information taken from the working memory and stored in an external memory is then referred to as cognitive trace. In the context of a diagram, a cognitive trace would be, for instance, to mark, update, and highlight information [19]. Likewise, in

the context of process variants, the model itself may serve as external memory. For example, when deriving a process variant in C-EPC for a particular context, the model reader may cross out model elements that have been removed (not requiring their storage anymore in the working memory). Rather, this information is transferred to the C-EPC model, freeing up working memory capacity.

Abstraction. Basically, the idea of abstraction is to hide information by aggregation. As irrelevant information can be hidden from the reader, it becomes easier to focus on relevant information, i.e., abstraction supports the human mind's attention management [12], leading to decreased mental effort. Unlike in C-EPC, where the process family is represented in a single model, Provop separates the base model from change options which are abstracted via variation points. This, in turn, simplifies both the base model and the change options, presumably making both easier to interpret, as attention is not distracted by an abundance of modeling elements.

Split-Attention Effect. Even though abstraction has been attributed to reduce mental effort [16, 20, 21], it typically co-occurs with the split-attention effect, which is known to increase mental effort [22]. In general, the split-attention effect occurs whenever information from different sources needs to be integrated. As the human mind can only focus on a single aspect at the same time [23], attention needs to be constantly switched between the information sources, leading to increased mental effort. In addition, the task of integrating information is known to further increase mental effort [22]. To illustrate the split-attention effect, consider the change options in the Provop approach. In this approach, the model reader has to constantly switch attention between the base model and the change options in order to extract a new process variant. In addition, the model reader has to integrate information from change options to the base model, i.e., adding, removing, moving model elements into the base model, further increasing the mental effort.

5 Qualitative Comparison

So far we have discussed C-EPC and Provop as representatives of approaches for modeling BP variability. In the following, we will use concepts from cognitive psychology to systematically assess differences between these two proposals with respect to understandability. Since understandability not only depends on the notation, but also on the type of task to be performed [6, 7], we have structured the comparison of both approaches along several such tasks. In particular, we will discuss the task of:

1. extracting a process variant from a configurable process model given a certain context,
2. understanding the factors driving variability (i.e., the context variables and their values), and
3. understanding the relationships between process fragments.

In our qualitative comparison, we assume a setting where the model reader has the models available in paper-based form. We are aware that, even though our discussion focuses on cognitive aspects only, tool support is indispensable for working with configurable models. However, to be able to develop effective tool support, it is essential to know what makes configurable process models hard to understand. Without a profound discussion, as provided here, tool development is rather driven by speculation than by systematic consideration.

In the following subsections, we provide a discussion, first for C-EPC and afterwards for Provop, structured along the following points: First, we will describe the task in the form of an algorithm describing the steps a model reader has to perform in order to perform the respective understandability task. The algorithms have been derived in an iterative manner by observing a set of model readers conducting the described tasks. Second, we will illustrate the presented algorithm using the screen postproduction project introduced in Section 2. Third, we will perform an in-depth analysis to determine the cognitive complexity of the task.

5.1 Extracting a Process Variant Suitable for a Certain Context

This section discusses first for C-EPC and then for Provop the extraction of a specific process variant for a given *context* from the configurable process model (i.e., domain characteristics of the intended process variant). Such a task is, for example, important to understand how the process looks like given a certain *context*. In addition, it is essential to determine whether a specific business requirement is covered (e.g., when creating or modifying the configurable process model). Extracting a process variant can also become necessary as part of a more complex task like understanding the commonalities/differences between a set of variants. To illustrate the process in both proposals, we assume that a model reader wants to derive the process variant that relates to the production of a low-budget project which implies the use of *tape* as medium for both shooting and delivery tasks.

5.1.1 Extracting a Process Variant Using C-EPC

To obtain a specific process variant from a configurable process model in C-EPC, the model reader has to inspect all configurable nodes (i.e., connectors and functions), evaluate the associated requirements, and configure them according to the given *context*. This process is explained in more detail in Algorithm 1.

Example. To obtain the process variant that relates to the low-budget project in C-EPC (i.e., when shooting and delivery medium is tape) the model reader starts with the configuration of *configurable connector 1*. For this purpose, the model reader locates all requirements attached to it and evaluates the ones of interest to extract the wanted process variant, i.e., reqs 1-3. According to the given context (i.e., shooting media is *tape*), *configurable connector 1* is configured as SEQ1_a as stated in req. 1. Next, the configuration of *configurable connector 2* has to be performed. In this case, req. 4 determines that the same

Algorithm 1 Algorithm for extracting a process variant from a C-EPC model

```
1: for all cn ∈ configurable nodes do
2:     Locate requirements associated to cn
3:     for all r ∈ associated requirements do
4:         Evaluate Boolean expression ex in r
5:         if r fits the given context then
6:             Configure cn as stated in r
7:             Break
8:         end if
9:     end for
10:    if no requirement fits or no requirements are attached then
11:        Change cn to a regular (non-configurable) node
12:    end if
13: end for
```

configuration performed to *configurable connector 1* should be applied to *configurable connector 2*, i.e., $SEQ2_a$ is chosen. Then, the configuration of *configurable connector 3* has to be done. For such purpose, reqs. 5–7 are evaluated. After evaluating req. 5 the model reader discovers that $SEQ3_a$ should be chosen. Unlike reqs. 1–3, reqs. 4–7 are entirely expressed in terms of the structure of the model (defining implications between alternative process fragments), not providing any information regarding the context driving the configuration of the variant. This means that the model reader, in order to understand why this configuration is performed, has to either remember the decisions previously taken or go back in the model and revisit the requirements that determined the configuration of related nodes. Similarly to the configuration performed for *configurable connector 2*, req. 8 determines that *configurable connector 4* should be configured equally to connector 3, i.e., as $SEQ4_a$. For configuring *configurable connector 5* req. 9–12 are evaluated. In this case, by evaluating req. 9 the model reader discovers that it should be configured as $SEQ5_a$ and that function *finish on tape* should be switched ON. The fact that these requirements include context variables help the model reader to better understand which configuration should be taken. Next, the model reader has to decide about the configuration of function *telecine transfer*. In this case req. 14 states that this function should be configured as OFF (since connector 3 was configured as $SEQ3_a$). Then, *configurable connector 6* is configured as $SEQ6_a$ according to req. 15. Finally, function *finish on new medium* is switched OFF since function *finish on tape* has been switched ON (req. 16).

Cognitive Discussion. Considering the cognitive complexity the use of C-EPC entails, we can identify three basic operations: locating elements (line 2), evaluating Boolean expressions (line 4), and adapting the model accordingly (line 6).

As argued in [24], the more distinct properties a visual element has, e.g., shape and color, the easier it is to identify (i.e., to *locate the element*). In C-EPC, requirements are represented by white tags, configurable connectors are represented by white circles with a thick border, whereas configurable functions

are represented by green rounded rectangles with a thick border. Hence, the reader can draw on two different properties (i.e., color and shape) for identifying distinct modeling constructs presumably requiring a low mental effort. For identifying whether there are requirements attached to a configurable node (line 2), the model reader can rely on pattern recognition [12] to efficiently perform this operation (requirements are connected via dotted lines).

The first real challenge occurs when the model reader has to *evaluate associated Boolean expressions*. As they can be arbitrarily complex and have to be interpreted in the model reader's mind, presumably a high mental effort can be expected. While some requirements can be evaluated by just considering their Boolean expression (i.e., req. 1–3 and 9–12), others depend on decisions previously taken (i.e., req. 4–8 and 13–16). This requires a bigger cognitive effort by the model reader, since the model reader has to remember decisions taken for already configured nodes. For example, when evaluating req. 14 for configuring function *telecine transfer*, the model reader has to go back to the related configurable nodes, i.e., to configurable connector 3. In addition, the model reader has to consider the configuration of functions *finish on tape* (i.e., req. 9, 10, and 12) and *finish on new medium* (i.e., req. 12) to understand the semantics of the configuration. The complexity of evaluating Boolean expressions can differ significantly depending on how many related requirements have to be additionally evaluated and the complexity of them.

Having evaluated the requirements, respective *model adaptations* have to be performed (e.g., removing model elements; cf. line 6). Even though these operations are rather simple the model reader has to keep track of all changes made during the configuration. Due to the limited nature of working memory (7 ± 2 slots), it seems essential that the model reader can offload parts of the working memory to an external memory, e.g., by annotating a print-out of the model.

5.1.2 Extracting a Process Variant Using Provop

To extract a process variant in Provop, the model reader has to proceed as follows. First, the model reader has to examine all change options and their Boolean expressions. Second, the model reader has—based on this evaluation—to select all change options whose context rule satisfies the given context. Third, the model reader has to determine for all the selected options, by examining the constraint model, whether they can be applied considering the already applied options. Fourth, the model reader has to locate variation points where options apply, and fifth, the model reader has to mentally integrate the change operations gathered in the selected change options into the base model. This process is explained in more detail in Algorithm 2.

Example. To extract a process variant in Provop the model reader first examines all change options including their associated context rules to determine which options are applicable for the given context (line 1–6). Based on this, the model reader then selects *options 1* and *5*, since only these two satisfy the given context (lines 3–4) and applies them to the based model (lines 7–14). For this the

Algorithm 2 Algorithm for extracting a process variant in Provop

```
1: for all o ∈ change options do
2:     Evaluate Boolean expression ex from associated context rule
3:     if ex satisfies the given context then
4:         Add o to the selected options list
5:     end if
6: end for
7: for all os ∈ selected options list do
8:     Evaluate constraints between all os
9:     if os is not excluded by any of the already applied options then
10:        for all ch ∈ change operations from os do
11:            Locate adjustment points in the base model as specified in ch
12:            Apply ch to the base model within the corresponding adjustment points
13:        end for
14:    end if
15: end for
```

model reader checks the constraints between the selected options (lines 8–9) and concludes that both *option 1* and *option 5* can be applied. The application of *option 1* involves deleting activity *prepare film for edit* between variation points A and B, and inserting activity *prepare tape for edit* instead (lines 11–12). In addition, activity *perform negmatching* is replaced by activity *edit footage online*. Moreover, the application of *option 5* implies the replacement of activity *finish on film* by *finish on tape* between adjustment points E and F (lines 11–12).

Cognitive Discussion. Considering the cognitive complexity of Provop we can identify two main operations. First, selecting appropriate change options, and second, applying these change options to the model.

For the *identification of relevant change options* the model reader inspects all change options and evaluates whether they are applicable for the current context, i.e., the model reader evaluates the Boolean expression associated with the change option (lines 1–6 in Algorithm 2). Similar to C-EPC, it can be expected that the interpretation of such Boolean expressions presumably imposes a high mental effort. However, unlike C-EPC, in Provop Boolean expressions are always expressed in terms of context variables. These variables provide semantics to the change options, helping the model reader to understand the intent of the options. After having identified relevant change options, the model reader has to apply them to the base model (cf. lines 7–14). Before applying a change option, it has to be checked whether the change option is conflicting with previously applied change options (lines 8–9). This task, however, can easily be accomplished using Provop's *option constraints*, i.e., a set of relationships (e.g., inclusion, exclusion) between *change options* targeted to ensure their proper use based on the semantics of the domain. As these option constraints are visually depicted, the model reader's recognition capabilities will help to efficiently identify conflicting options, hence presumably imposing a low mental effort.

Regarding the actual *manipulation of the model* (lines 11–12), the effort for integrating the change options into the base model is determined by the *change*

distance [25] between the base model and the variant to be derived. In other words, the more modeling elements are added to / removed from the base model, the more complex the integration task will be. In addition, the type of change operations contained in the change options influences complexity. For example, when deleting an activity, respective parts can be removed from the model by hiding them with a finger on the print-out of the model, or by crossing them out using a pen. In this way, the model reader does no longer have to keep this information in his working memory. Rather, the model serves as external memory, freeing up mental resources. When inserting activities, in turn, this possibility is not available (except this is done through a supporting modeling tool) and has to be done in the reader's mind. Therefore, complexity grows with the number of changes applied to the base model. Therefore, an optimized design of the base model that requires minimum changes to derive different variants presumably requires less effort by the model reader [39]. Altogether it can be said that most mental effort will presumably be required for evaluating Boolean expressions as well as conducting model changes.

5.1.3 Discussion

After studying both proposals three major differences can be observed.

First, in a C-EPC model, modeling elements are mainly removed from the configurable model (with exception of functions when these are configured as optional, which involve the inclusion of some branching condition in the model). By contrast, in Provop model elements can either be added, deleted, or moved during the configuration process. As argued above, the cognitive complexity depends on the type of change operations to be performed (i.e., deletion operations presumably involve less cognitive effort than insertions or movements).

Second, requirements and configurable nodes are integrated in C-EPC, whereas change options and the base model are separated in Provop. Similar to [20, 21], we argue that for small models, C-EPC presumably is easier to understand, as all the information is integrated and hence in contrast to Provop no split-attention effect can be expected. However, when model size increases, models may quickly become too complex resulting in an overload for the model reader, especially when there are many relationships between alternative modeling elements. Here, it can be assumed that the abstraction mechanisms provided in Provop (i.e., represented by change operations defined separately from the base model) contributes to retain understandability even for large models.

Third, even though Boolean expressions need to be evaluated in both approaches, the way they are used by the two proposals differs. In C-EPC, one of the biggest challenges faced by the model reader relates to the fact that alternatives are usually expressed at the structural level, neglecting the semantics associated to the different alternatives. This fact involves that, in some cases, the model reader has to evaluate the Boolean expression at hand, but additionally has to keep track of previously made decisions. In contrast, in Provop, Boolean expressions are always expressed in terms of context variables, which contribute to better understand the semantics of the associated change operations. In ad-

dition, the concept of options (i.e., grouping of related change operations) as provided by Provop and the explicit specification of constraints between them in the constraint model presumably reduces the mental effort required by the reader for understanding them. Hence, from this point of view, Provop models presumably impose a lower mental effort on average.

5.2 Identifying Context Variables

This section discusses first for C-EPC and then for Provop how the factors that lead to variations (i.e., the context variables including their values) can be identified in a configurable process model. Context variables provide model readers with the required contextual information to understand the reasons for variations (e.g., different procedures are followed depending on the *shooting_media* as well as the *delivery_media*). Context variables also play an important role in identifying the parts of a configurable process model that need to be changed (e.g., when a new delivery medium is added). In addition, context information is also essential when trying to understand differences between variants. Moreover, the different values that a certain context variable can take helps to assess whether the configurable process model covers all possible alternatives or not (e.g., whether *tape*, *film*, and *tape and film* are considered as valid *shooting media*). To illustrate the process in both proposals we consider the task of extracting all context variables and all their allowed values from their respective modeling artefacts.

5.2.1 Identifying Context Variables in C-EPC

To extract context related information from the configurable process model in C-EPC, the model reader has to locate all the requirements associated with the configurable elements (i.e., nodes and functions) included in the configurable process model. Then, based on the Boolean expressions included in these requirements, the model reader has to identify the context variables and their allowed values. This process is explained in more detail in Algorithm 3.[2]

Example. First of all, the model reader checks requirement 1 and from its logic predicate extracts context variable *shooting_media* and its first value which is *tape*. When examining requirement 2 the model reader realizes that context variable *shooting_media* has already been used in requirement 1. However, a new valid value for this variable can be extracted, i.e., *film*. Similarly, when evaluating requirement 3 a new value for *shooting_media* is found, i.e., *tape and film*. On the contrary, the examination of requirements 4–8 does neither yield any additional context variable nor any additional value for the already identified variables. By looking at requirement 9, a new context variable (i.e., *delivery_media*) is discovered and also a first value for it (i.e., *tape*). The examination of requirements 10

[2] Note that we assume that context variables are included in the requirements (cf. Section 3.1 for details). When using the questionnaire approach together with C-EPC, in turn, the algorithm looks slightly different, since context variables are modeled explicitly in the questionnaire and can be extracted from there.

Algorithm 3 Algorithm for extracting process context information in C-EPC

```
1: for all r ∈ requirements associated with configurable elements do
2:     Identify context variables cv in r
3:     for all cv ∈ context variables in r do
4:         if cv is not yet included in the list of context variables then
5:             Add cv to the context variable list
6:         end if
7:         Identify values for cv
8:         for all v ∈ values for cv do
9:             if v is not yet included in the list of allowed values for cv then
10:                Add v to the allowed values for cv
11:            end if
12:        end for
13:    end for
14: end for
```

and 12 shows that three new values for the *delivery_media* variable context are also valid, i.e., *film*, *tape and film*, and *new medium*. Finally, requirements 13–16 refer to the associated functions and define whether these should be included, skipped, or optionally included in the model. However, they define neither a new context variable nor a new value for existing ones.

Cognitive Discussion. From a cognitive point of view, the algorithm can be analyzed as follows. The identification of requirements, i.e., the identification of shapes, is performed by recognition (line 1). Similarly, in line 2, the context variables have to be identified. As illustrated in Fig. 1, context variables have the following form: *context_variable* = *value*, e.g., *shooting_media* = *'FILM'*. Again, we argue that this identification can mainly be performed by recognition, which presumably imposes a low mental effort. In lines 4–6, the model reader has to evaluate whether the variable is already known. Even though the comparison with existing variables itself can be considered to be rather easy, the 7±2 slots of working memory available will most likely not be enough to perform this task for a non-trivial model without the support of some kind of external memory. Analogously, the extraction of values in lines 7–12 starts with the identification of values, i.e., recognition. Then, the model reader has to check whether a value has already been found and add the value to the list of determined values, if not. Also here we argue that this mental process itself is rather simple, as it merely includes the comparison of values. However, intermediate results have to be stored, hence the model reader will most likely need to use external memory to perform this task.

5.2.2 Identifying Context Variables in Provop

For the extraction of context variables the model reader can directly go to the context model where all context variables and their allowed values are defined (i.e., table with all context variables + possible values; cf. bottom left part of Fig. 2). Algorithm 4 describes this process in more detail.

Algorithm 4 Algorithm for extracting process context information in Provop

```
1: for all cv ∈ context model do
2:     Read cv and allowed values
3: end for
```

Example. In this case just by looking at the context model the model reader knows that there are just two context variables which are *shooting_media* and *delivery_media*. While the former can be valued as *tape*, *film*, or *tape and film* the latter can be valued as *tape*, *film*, *tape and film*, or *new_media*.

Cognitive Discussion. As discussed, in Provop context variables are explicitly modeled in the context model. This information can directly be extracted from the model and therefore requires nothing but reading the context values and allowed values—a low cognitive load can be assumed for this task.

5.2.3 Discussion

As it is explained in this section, for C-EPC models, the extraction of the context variables and their values is only implicitly available and thus has to be computed by going through all requirements. On the contrary, in Provop, this information is explicitly represented in the context model, which facilitates context variable identification.

5.3 Identifying Relationships between Variable Process Fragments along a Process Variant

This section deals with the question whether certain relationships (e.g., inclusion or exclusion) between variable process fragments exist. The identification of these relationships is very important to ensure that no inconsistencies in respect to the domain semantics exist in the process model family; these relationships establish conditions of use of the process fragments when a process variant is derived. To illustrate how these conditions can be checked we assume that we want to ensure that *telecine transfer* is only performed when the *shooting_media* is *film* or *tape and film* and the *delivery_media* is either *tape*, *tape and film* or *new medium*.

5.3.1 Identifying Relationships between Variable Process Fragments in C-EPC

To accomplish this comprehension task in C-EPC, the model reader has to locate and select all the requirements that are related (either directly or indirectly) to the relationship constraint being checked. Then, for each selected requirement, the model reader has to evaluate the associated Boolean expression and check whether this satisfy or not the constraint relationship. Algorithm 5 defines this process more in detail.

Example. According to the example, in C-EPC the model reader has to locate all the requirements that refer to function *telecine transfer*, and to the

Algorithm 5 Algorithm for identifying relationships between process fragment alternatives in C-EPC

```
1: for all r ∈ requirements do
2:     if r has an impact (directly or indirectly) on the condition then
3:         Add r to collected requirements list
4:     end if
5: end for
6: Analyze relationships between fragments in collected requirements list to decide
   whether the relationship constraint is satisfied or not
```

context variables *shooting_media* (valued as *film*) and *delivery_media* (valued as *tape* or *new medium*). First of all, the requirement providing the configuration for function *telecine transfer* (i.e., requirement 14) states that this function is only switched on if configurable connector 3 has been configured either as $SEQ3_b$ or as an AND and when functions *finish on tape* and *finish on new medium* are switched on. Regarding the configuration of connector 3, the model reader can see the configurations to $SEQ3_b$ or AND are given by requirements 6 and 7 respectively. In turn, these requirements point to the configuration of configurable connector 1 and require that is is configured to either $SEQ1_b$ or AND. These configurations are given by requirements 2 and 3. Analyzing these requirements reveals that respective configurations can be obtained if *shooting_media* is *film* (i.e., req. 2) or *shooting_media* is *tape and film* (i.e., req. 3). At this stage the model reader already knows that function *telecine transfer* is only performed if *shooting_media* is *film* or *tape and film*, but not if shooting media is *tape*. Then, the model reader has to check whether the requirements also enforce that the *delivery_media* is either *tape* or *new medium*, but not *film* or *tape and film*. For this purpose, the model reader has to check requirements req. 9, 10, and 12. All three requirements configure configurable connector 5 in such a way that either $SEQ5_a$ or AND are chosen and function *telecine transfer* can be reached. In addition, they ensure that functions *finish on tape* or *finish on new medium* are switched ON, which is a requirement for function *telecine transfer* to be switched ON. As a result, after checking all the involved requirements, the model reader can conclude that *telecine transfer* is only performed when shooting medium is *film* or *tape and film* and delivery medium is *tape*, *tape and film*, or *new medium*.

Cognitive Discussion. This task, as described in Algorithm 5, can be separated into two parts. First, the model reader has to locate the set of requirements involved in the relationship constraint being checked (cf. lines 1–5). In the second part, the model reader determines whether the selected requirements ensure that this constraint is well represented. Locating the requirements involves checking all the requirements attached to the process fragments involved in the relationship constraint being evaluated. In our example, these requirements are 14 (for function *telecine transfer*), requirements 6 and 7 (for connector 3), requirements 1 and 3 (for context variable *shooting_media*), and requirements 9, 10 and 12 (for context variable *delivery_media*). While locating requirements constitutes a relatively simple task, analyzing all the involved requirements (i.e., req. *r1*, *r2*,

$\ldots r_n$) to finally find out if the relationship to be checked is well represented in the model (i.e., to determine if there is a solution to the Boolean formula $r1 \wedge r2 \wedge \ldots \wedge r_n$), presumably requires high mental effort. The complexity of this task significantly depends on the number of requirements that have to be evaluated as well as their complexity. In principle, this problem is known to be np-complete [38] and therefore only possible to be done in one's mind for relatively small problems.

Hence, we conclude that for this task the step 1 (identification of requirements) may be done manually, but for step 2 (satisfying all requirements), tool support is indispensable for models that go beyond the complexity of toy examples.

5.3.2 Identifying Relationships between Variable Process Fragments in Provop

In Provop relationships between process fragments can be found at two different levels of abstraction. On the one hand, change options allows explicitly defining inclusion and exclusion relationships by means of delete and insert operation respectively. On the other hand, option constraints allow again defining relationships, but this time at a higher level of abstraction, i.e., at the options level. Therefore, the model reader has to analyze both, change options and the option constraints to find out about relationships between process fragments. Algorithm 6 defines this process in more detail.

Algorithm 6 Algorithm for identifying relationships between process fragment alternatives in Provop

```
1: for all o ∈ change options do
2:     if o refers to the fragments involved in the condition then
3:         Add o to collected change options list
4:     end if
5: end for
6: Analyze relationships between fragments in change options list to decide whether
   the relationship constraint is satisfied or not
```

Example 1. To test that *telecine transfer* is only performed when the *shooting_media* is *film* or *tape and film* and the *delivery_media* is either *tape*, *tape and film* or *new medium*, the the model reader first has to locate the change options that insert activity *telecine transfer* (since this activity is not part of the base model). In the example, change option 5 is the only option inserting activity *telecine transfer*. Since there is no further option inserting activity *telecine transfer* the question can be answered locally by looking at the context rule associated with option 5. Evaluating the respective Boolean expression shows that activity *telecine transfer* is only inserted in the base model if the *shooting_media* is *film* or *tape and film* and the *delivery_media* is *tape*, *tape and film* or *new medium*.

In Provop, the complexity for identifying relationships between variable process fragments depend on how explicit these relationships are made in the model. In example 1, the complexity is low since all the information needed can be found in the change option 5. However, as examples 2 and 3 show, this complexity can grow when the information is not so explicit in the model.

Example 2. Testing, in turn, whether activity *finish on tape* and *finish on new medium* can co-occur is slightly more complex, since this task cannot be answered locally by looking at one option only. Like in the previous example we start by locating all change options where activity *finish on tape* or *finish on new medium* are inserted (i.e., options 5, 6, and 7). Since there is no single option with both activities we cannot yet decide, but have to analyze the situation further to determine whether it is possible to execute options 5 and 7 together or options 6 and 7 (in both cases this would result into a co-occurrence of activities *finish on tape* and *finish on new medium*). For this, we first consult the option constraint model and see that option 5 excludes option 7 and option 6 excludes option 7 as well. Based on this information we can conclude that the two activities cannot co-occur. If these relationships would not be defined explicitly in the constraint model, we would have to check the context rules of options 5, 6, and 7 and check whether there is a possibility that options 5 and 7 as well as options 6 and 7 co-occur.

Example 3. To answer the question whether a variant exists for which activity *telecine transfer* and *record digital film master* co-occur is even more complex. This task cannot be answered locally by looking at one option only or using the constraint model, but requires an analysis of the associated context rules, since the relationships between these two activities are only implicitly defined. Like in the previous example we start by locating all change options where activity *telecine transfer* or activity *record digital film master* are inserted (i.e., options 3 and 4). Since the question whether these two activities co-occur cannot be answered by looking at the change options individually, the model reader checks in the constraint model whether there are any constraints between option 3 and option 4. Since options 3 and 4 are not mutually exclusive based on the constraint model, the associated context rules have to be analyzed in detail. This analysis shows that activities *telecine transfer* and *record digital film master* cannot co-occur. *Telecine transfer* requires *shooting media* to be *film* or *tape and film*, while *record digital film master* is only executed if *shooting media* is *tape*.

Cognitive Discussion. Similar to the algorithm for C-EPCs, Algorithm 6 can be broken down into two main parts. First, the model reader *identifies relevant change options* in lines 1–5. Second, the model reader checks whether the collected change options *satisfy the relationships* to be tested. For identifying all related change options, the model reader has to inspect each change option individually as no change option can be excluded beforehand. In our example, all change options that insert activity *telecine transfer* have to be considered.

Having identified the relevant change options the relationships between them have to be analyzed (cf. line 6). Depending on the type of question asked, the decision of whether or not the identified *change options satisfy the relationships*

can be made based on a single option (e.g., option 3 in Example 1), by analyzing the constraint model (e.g., Example 2), or might require analyzing the context rules associated with the identified change options (e.g., Example 3). If the question cannot be decided locally based on a single option or based on the option constraints, like for a C-EPC model, a solution has to be found that satisfies the relationship to be tested. Hence, in general, this task can be considered np-complete for Provop as well, unless shortcuts are possible.

5.3.3 Discussion

After studying both proposals for identifying relationships between variable process fragments, one main point can be observed. In both proposals, it is necessary to go through all requirements and change options, respectively, to find the ones referring to the condition being evaluated. Then, in C-EPC the model reader has to infer the type of existing relationship between the fragments. In turn, in Provop, these relationships are potentially explicitly defined as part of a change option or in the constraint model, which facilitates the analysis. If not, similarly to C-EPC, the model reader has to evaluate the associated context rules to find out whether the condition being evaluated is satisfied or not. Put differently, if the option constraints can be used to determine the relationships between variable process fragments or the relationships can be answered by a single change option locally, then the task can be facilitated tremendously. Otherwise it will presumably be, just as for C-EPC, extremely difficult to accomplish this tasks without tool support.

6 Related Work

Within the business process management community, several proposals have been developed to deal with the representation of variability in business process models (e.g., Provop [1], C-EPC [2], PESOA [3], Rule representation and processing [4], and PPM [5]). These works take a design-oriented perspective and provide technical solutions for managing variability; the understandability of the artifacts created using such approaches is not in the focus. Recently qualitative evaluations of the C-EPC proposal in form of case studies have been conducted [26, 27]. However, to the best of our knowledge no work has addressed understandability of process model families explicitly. Closely related to this is existing research on the understandability of process models. For example, in [33], understandability of process models is approached from a theoretical point of view. Complementary, several studies report on empirical investigations. Most approaches thereby employ the concept of metrics computed on structural aspects of the process model to assess understandability, e.g., [28, 29, 30, 37, 34]. While metrics seem to be a promising approach to assess model complexity and understandability, in [6, 7] it is shown that understandability of a process model significantly depends on the type of question asked. Consequently, a metric will only be able to roughly estimate understandability. In [31, 20, 21], concepts from

cognitive psychology are used as a tool to discuss understandability of process models for specific comprehension tasks in the context of both imperative and declarative process models. In this paper we build upon this work, and extend it to discuss a selected comprehension task specific to process model families. Another interesting work to be mentioned is [35], in which empirically validated guidelines for process modeling, with the aim to improve understandability, are presented. Unlike in this paper, aspects specifically related to the understanding of process model families are not addressed.

7 Summary and Outlook

The main goal of this paper is to compare the structural and behavioral approaches for modeling process model families in terms of understandability. Instead of looking at understandability from a broad perspective, the discussion is centered around the extraction task of a process variant from the modeling artifacts produced by C-EPC and Provop. The different approaches are discussed in terms of different concepts from cognitive psychology based on which the mental effort required to understand the modeling artifacts produced by the two approaches can be estimated. In turn, this allows us to estimate the understandability of the two approaches for specific comprehension task. The task addressed in this paper constitutes just a first attempt regarding the investigation of understandability of these two approaches. Formal metrics and experiments involving a large number of subjects and different configurable process models are planned as future work to empirically test the discussion. Based on the comprehension tasks presented in this paper, we will conduct a series of experiments to empirically assess the understandability of both approaches and to investigate the factors that impact understandability of process model families improving the modeling of such families and facilitating their reuse. For the conduction of such experiments, we will use the Cheetah Experimental Platform [40] which not only allows testing the outcome of process modeling (i.e., the created process models), but also the process of process modeling itself.

References

1. Hallerbach, A., Bauer, T., Reichert, M.: Capturing variability in business process models: the Provop approach, J. Soft. Maintenance. 22(6–7):519–546 (2010)
2. Rosemann, M., van der Aalst, W.M.P.: A configurable reference modeling language. Inf. Systems 32(1):1–23 (2007)
3. Puhlmann, F., Schnieders, A., Weiland, J., Weske, M.: Variability mechanisms for process models. Technical report, BMBF–Project (2006).
4. Kumar, A., Wen, Y.: Design and management of exible process variants using templates and rules. Int. J. Comput. Ind. 63(2), pp. 112–130 (2012).
5. Pascalau, E., Awad, A., Sakr, S., Weske, M.: On Maintaining Consistency of Process Model Variants. BPM Workshops, 289-300 (2010).

6. Figl, K., Laue, R.: Cognitive Complexity in Business Process Modeling. In Proc. CAiSE'11, 452–466.
7. Melcher, J., Detlef, S.: Towards Validating Prediction Systems for Process Understandability: Measuring Process Understandability. In Proc. SYNASC'08, 564–571.
8. La Rosa, M., Lux, J., Seidel, S., Dumas, M., ter Hofstede, A.H.M.: Questionnaire-driven Configuration of Reference Process Models. In Proc. CAiSE'07, 424-438.
9. van der Aalst, W.M.P., Dreiling, A., Gottschalk, F., Rosemann, M., Jansen-Vullers, M.H.: Configurable process models as a basis for reference modeling. In BPM Workshops, LNCS vol. 3812, 512–518 (2005).
10. Vervuurt, M.: Modeling business process variability: a search for innovative solutions to business process variability modeling problems". Student Theses of University of Twente. October 2007.
11. Hallerbach, H., Bauer, T., Reichert, M.: Issues in Modeling Process Variants with Provop. Business Process Management Workshops, 2008, pp. 56–67 (2008)
12. Larkin, J.H., Simon, H.A.: Why a Diagram is (Sometimes) Worth Ten Thousand Words. Cognitive Science, 11(1):65–100 (1987).
13. Tracz, W. J.: Computer programming and the human thought process. Software: Practice and Experience, 9(2):127–137 (1979).
14. Paas, F., Tuovinen, J.E., Tabbers, H., Van Gerven, P.W.M.: Cognitive Load Measurement as a Means to Advance Cognitive Load Theory. Educational Psychologist, 38(1):63–71 (2003).
15. Miller, G.: The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. The Psychological Review, 63(2):81–97 (1956).
16. Moody, D. L.: Cognitive Load Effects on End User Understanding of Conceptual Models: An Experimental Analysis. In Proc. ADBIS'04, 129–143.
17. Sweller, J.: Cognitive load during problem solving: Effects on learning. Cognitive Science, 12(2):257–285 (1988).
18. Zugal, S., Pinggera, J., Weber, B.: The Impact of Testcases on the Maintainability of Declarative Process Models. In Proc. BPMDS'11, 163–177.
19. Scaife, M., Rogers, Y.: External cognition: how do graphical representations work? Int. J. Human-Computer Studies, 45(2):185–213 (1996).
20. Zugal, S., Pinggera, J., Mendling, J., Reijers, H.A., Weber, B.: Assessing the Impact of Hierarchy on Model Understandability-A Cognitive Perspective. In Proc. EESSMod'11, 123–133.
21. Zugal, S., Soffer, P., Pinggera, J., Weber, B.: Expressiveness and Understandability Considerations of Hierarchy in Declarative Business Process Models. In Proc. BPMDS'12, 167–181.
22. Sweller, J., Chandler, P.: Why Some Material Is Difficult to Learn. Cognition and Instruction, 12(3):185–233 (1994).
23. Feldmann Barrett, L., Tugade, M. M., Engle, R. W.: Individual Differences in Working Memory Capacity and Dual-Process Theories of the Mind, Psychol Bull. 130(4):553-573 (2004).
24. Moody, D.L.: The "Physics" of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. IEEE Trans. Soft. Eng. 35(6):756–779 (2009).
25. Li, C., Reichert, M., Wombacher, A.: On Measuring Process Model Similarity Based on High-Level Change Operations. In Proc. ER'08, 248–264.
26. Lönn, C.M., Uppström, E., Wohed, P., Juell-Skielse, G.: Configurable Process Models for the Swedish Public Sector. In Proc. CAiSE'12, 190–205.

27. Gottschalk, F., Wagemakers, T., Jansen-Vullers, M., van der Aalst, W., La Rosa, M., van Eck, P., Gordijn, J., Wieringa, R.: Configurable Process Models: Experiences from a Municipality Case Study. In Proc. CAiSE'09, 486–500.
28. Mendling, J., Reijers, H.S., Cardoso. J.:What Makes Process Models Understandable? In Proc. BPM'07, 48–63.
29. Vanderfeesten, I., Reijers, H.A., van der Aalst, W.M.P.: Evaluating workflow process designs using cohesion and coupling metrics. Int. J. Comput. Ind. 59(5):420–437 (2008).
30. Reijers, H.A., Mendling, J.: A Study into the Factors that Influence the Understandability of Business Process Models. SMCA 41(3):449–462 (2011).
31. Zugal, S., Pinggera, J., Weber, B.: Assessing Process Models with Cognitive Psychology. In Proc. EMISA'11, 177–182.
32. La Rosa, M., van der Aalst, W.M.P., Dumas, M., ter Hofstede, A.H.M.: Questionnaire-based variability modeling for system conguration. Software and System Modeling 8(2), pp. 251–274 (2009).
33. Becker, J., Rosemann, M., Uthmann, C.: Guidelines of Business Process Modeling. In Proc. BPM'00, 30–49 (2000).
34. Melcher, J., Mendling, J., Reijers, H.A., Seese, D.: On Measuring the Understandability of Process Models. In Proc. BPM Workshops, 465–476 (2009).
35. Mendling, J., Reijers, H.A., van der Aalst, W.M.P.: Seven process modeling guidelines (7PMG). Information & Software Technology, 52(2):127-136 (2010).
36. Bannert, M.: Managing cognitive load–recent trends in cognitive load theory. Learning and Instruction, 12(1):139–146 (2002).
37. Cardoso, J.: Process control-flow complexity metric: An empirical validation. In Proc. IEEE SCC'06, 167–173 (2006).
38. Cook, S.: The complexity of theorem proving procedures. In Proc. STOC'71. 151-158 (1971).
39. Li, C., Reichert, M., Wombacher, A.: Mining business process variants: Challenges, scenarios, algorithms. Data Knowl. Eng. 70(5): 409–434 (2011).
40. Pinggera, J., Zugal, S., Weber, B.: Investigating the process of process modeling with Cheetah Experimental Platform. In Proc. ER-POIS'10, 13–18 (2010).